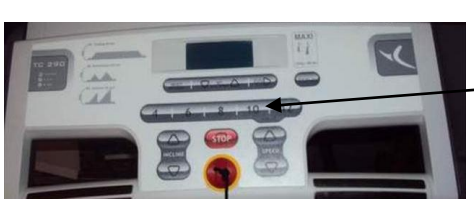
Décrire et analyser le comportement d'un système

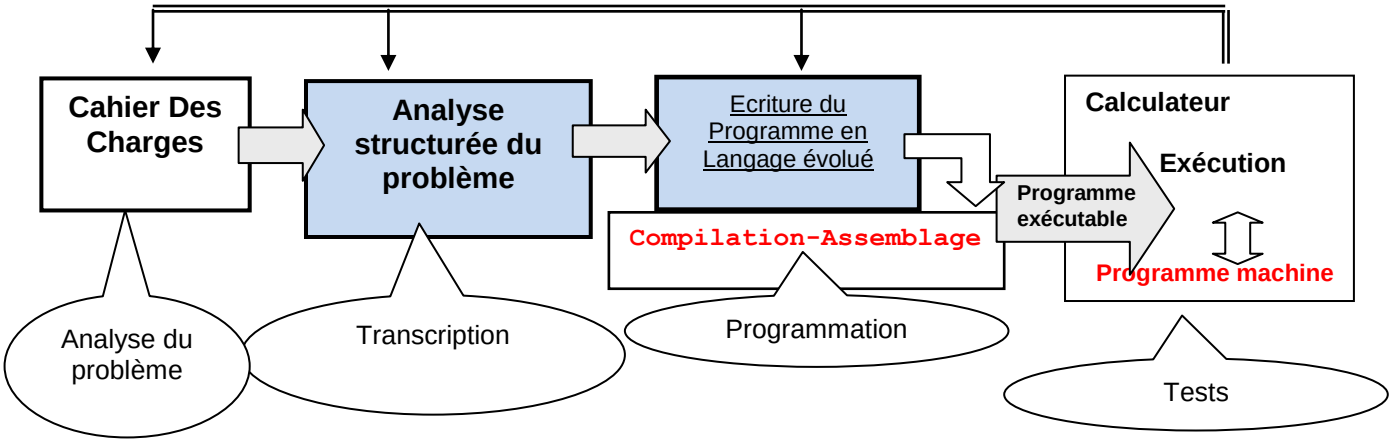
Cours

1. Mise en situation :

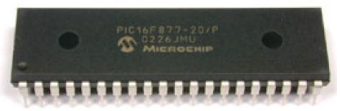


Problématique :
Gérer l'avance et l'inclinaison du tapis en fonction de touches sur son pupitre de commande.

Comment traduire ce cahier des charges, quel composant et quel langage utiliser ?

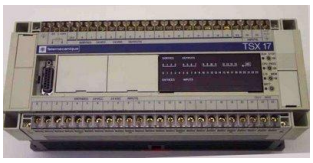


2. Choix du composant



Un **microcontrôleur** (µC) est un composant programmable qui intègre dans un même boîtier :

- Un microprocesseur (CPU Central Processing Unit ou µP) qui traite des données logiques et arithmétiques et réalise le séquencement des opérations
- Une mémoire de programme
- Une mémoire de données



Un **automate programmable industriel**, ou **API**, est un dispositif électronique programmable destiné à la commande de processus industriels par un traitement séquentiel. Il est structuré autour d'un microprocesseur (CPU), d'une alimentation par des sources de tension alternative (AC) ou continue (DC) et d'une mémoire.

Des interfaces d'entrée – sortie permettent de communiquer avec des périphériques externes.

3. Outils de description

Comment décrire le comportement séquentiel d'un système ?

La description du comportement d'un système séquentiel peut être réalisée notamment par :
- L'outil algorithme (ou algorigramme) ;
- L'outil graphe d'états ;
- L'outil grafcet (cas particulier).

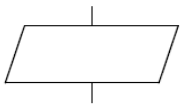
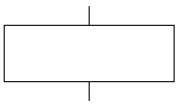
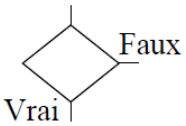
Il s'agit essentiellement d'outils graphiques permettant de modéliser le comportement séquentiel en termes de déroulement d'actions temporelles.

3.1 L'algorithme

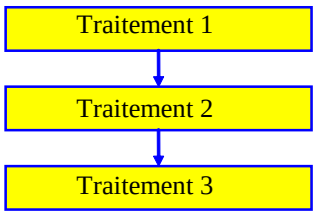
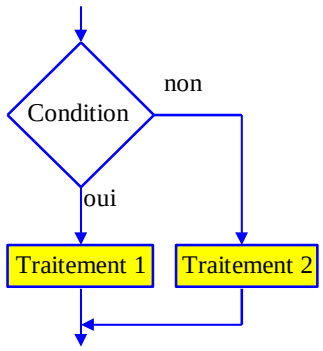
Un algorithme est un modèle universel de description d'un système numérique. Il existe deux types de représentation : **« l'organigramme » et « une représentation littérale algorithmique ».**

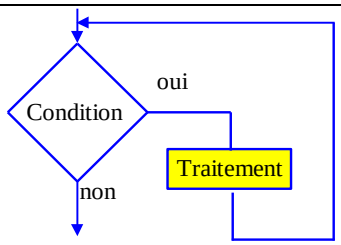
3.1.1 L'organigramme d'un programme

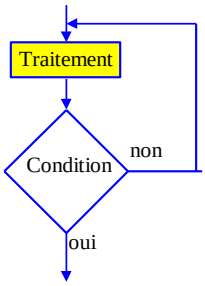
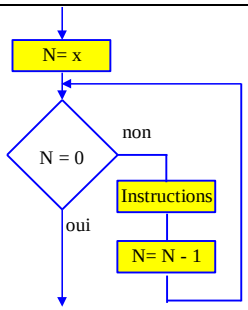
Un organigramme est la représentation graphique d'une suite structurée d'instructions. Voici les symboles graphiques normalisés à utiliser :

Symbole	Désignation	Symbole	Désignation
	**Début, fin, interruption** Début, fin ou interruption d'un programme		**Lecture ou Ecriture d'une donnée externe** Mise à disposition d'une information (écriture sur un port de sortie) ou enregistrement d'une information (lecture d'un port d'entrée).
	**Traitement interne** Opération ou calcul sur des données dont le résultat est stocké dans le microcontrôleur.		
	**Sous-programme** Portion de programme considérée comme une simple opération.		**Branchement** Test, exploitation de conditions variables impliquant le choix d'une parmi deux. Symbole utilisé pour représenter une décision.

3.1.2 Les structures algorithmiques

Structure	Organigramme	Algorithme
Séquence linéaire La structure linéaire se caractérise par une suite d'actions à exécuter successivement dans l'ordre de leur énoncé.		**FAIRE « traitement 1 »** **FAIRE « traitement 2 »** **FAIRE « traitement 3 »**
Séquence alternative ou conditionnelle Une structure alternative n'offre que deux issues possibles s'excluant mutuellement. Les structures alternatives définissent une fonction de choix ou de sélection entre l'exécution de l'un ou de l'autre des deux traitements.		**SI « condition » vraie** **ALORS FAIRE** **« traitement 1 »** **SINON FAIRE** **« traitement 2 »** **FIN SI**

Séquence répétitive ou itérative (boucle avec pré-test) Dans cette structure on commence par tester la condition, si elle est vraie alors le traitement est exécuté.		**TANT QUE « condition » vraie** **FAIRE « traitement »** **FIN TANT QUE**

Structure	Organigramme	Algorithme
Séquence répétitive ou itérative (boucle avec post-test) Dans cette structure le traitement est exécuté une première fois puis sa répétition se poursuit jusqu'à ce que la condition soit vérifiée.		**REPETER** **« traitement »** **JUSQU'A « condition » vraie**
Boucle avec comptage		**POUR N = x A 0 REPETER** **« traitement »** **FIN POUR**

3.2 Le graphe d'état

Les états peuvent être représentés graphiquement soit par des rectangles, soit par des ovales. Lorsqu'un état est composé de sous-états, on parle de super-état ou d'état composite.

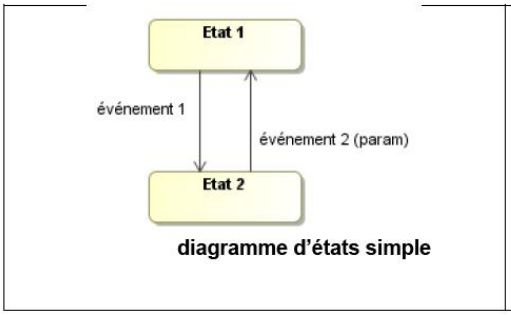
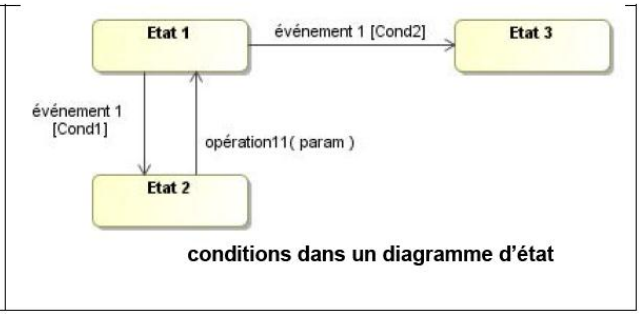
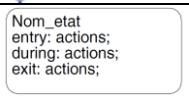


diagramme d'états simple



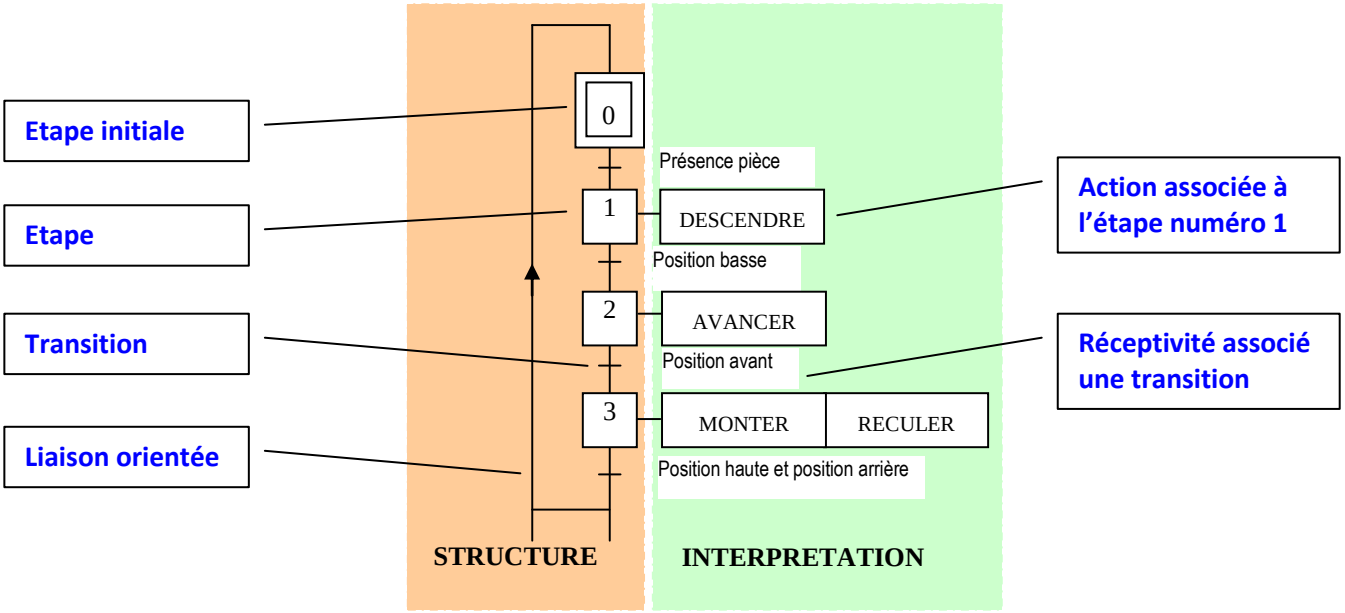
conditions dans un diagramme d'état

	**Etat OU :** Les états OU représentent des états de fonctionnement mutuellement exclusif. Des états OU ne peuvent pas être actifs ou s'exécuter en même temps. Il s'agit donc de la fonction Ou Exclusif. On parle alors d'états exclusifs, et on associe un nom à chaque état.
	**Etat ET :** Les états ET représentent des états de fonctionnement totalement indépendants. Plusieurs états de même niveau hiérarchique peuvent être actifs simultanément. Ces états sont représentés graphiquement par un rectangle en trait pointillé, et un numéro indique l'ordre d'exécution. On associe un nom à chaque état.
	**Transitions :** Les transitions sont représentées par des flèches orientées, et permettent de décrire les évolutions du système d'un état source vers un état destination.

	**Transition par défaut :** Cette transition indique l'état (ou super-état) qui doit être actif à l'état initial (« mise sous tension »).
	**Actions dans un état :** Il s'agit de définir les actions à effectuer lorsque l'état *Nom-etat* est actif. On définit 3 types d'actions : - Action à l'activation de l'état : Pour spécifier ce type d'action, la syntaxe est *entry*: *actions*. - Action durant l'état : Pour spécifier ce type d'action, la syntaxe est *during*: *actions*. Généralement, l'écriture de *actions* suffit. - Action à la désactivation de l'état : Pour spécifier ce type d'action, la syntaxe est *exit*: *actions.*

3.3 Le grafcet

Le GRAFCET est un outil graphique de description des comportements d'un système logique séquentiel. Il est très utilisé pour la programmation des automates programmables industriels (API). **Il est composé d'étapes, de transitions et de liaisons :**



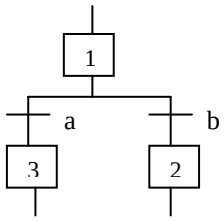
Une LIAISON est un arc orienté (ne peut être parcouru que dans un sens). A une extrémité d'une liaison il y a une (et une seule) étape, à l'autre une transition.

Une ETAPE correspond à une phase durant laquelle on effectue une ACTION pendant une certaine durée. On numérote chaque étape par un entier positif, mais pas nécessairement croissant par pas de 1, il faut simplement que jamais deux étapes différentes n'aient le même numéro.
Une étape est dite active lorsqu'elle correspond à une phase "en fonctionnement", c'est à dire qu'elle effectue l'action qui lui est associée. On représente quelquefois une étape active à un instant donné en dessinant un point à l'intérieur.

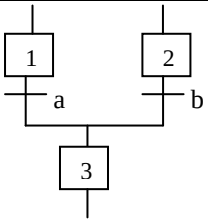
Une TRANSITION est une condition de passage d'une étape à une autre. Elle n'est que logique (dans son sens Vrai ou Faux), sans notion de durée. **La condition est définie par une RECEPTIVITE qui est** généralement une expression booléenne (c.à.d avec des ET et des OU) de l'état des capteurs.

Divergence en OU (structure alternative) :

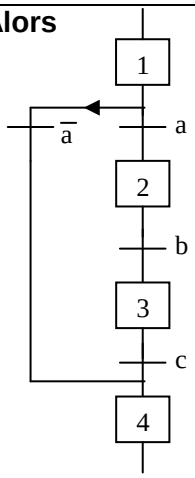
La structure linéaire se caractérise par une suite d'actions à exécuter successivement dans l'ordre de leur énoncé.



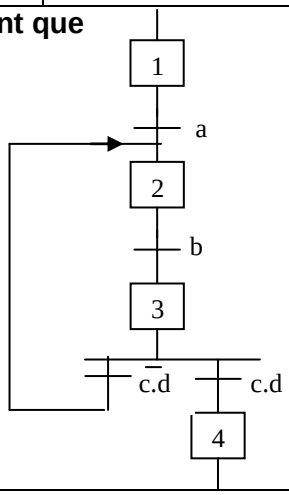
Convergence en OU (structure alternative)
Si 1 active et a sans b, alors activation de 3 et désactivation de 1, 2 reste inchangé
Si 1 et 2 et a et b alors 3 seule active



Boucle Si Alors



Boucle Répéter Tant que



4. Traduction dans un langage de « programmation »

Les mots-clés ou les symboles graphiques sont remplacés par les mots appartenant à la syntaxe du langage utilisé. Cette dernière étape devrait être celle à laquelle le concepteur consacre **le moins de temps** (dans l'hypothèse où les deux étapes précédentes ont été correctement développées !)

Il existe des langages de **haut niveau** : Java, C++, Basic, VPL…

Et des langages de bas niveau : assembleur

5. Organisation d'un algorithme (et d'un programme)

5.1 L'en tête

Dans cette partie le concepteur donne un **nom** à l'algorithme. Il définit le traitement effectué et les données auxquelles il se rapporte.

5.2 La partie déclarative

Dans cette partie, le concepteur décrit les différents « **objets** » que l'algorithme utilise.

5.2.1 Les constantes

Ce sont des « objets » constants dans tout l'algorithme.

Déclaration

Nom_Constante = valeur;

Exemple : **Constantes** Pi = 3,1416;

La déclaration de constantes symboliques permet de donner un nom à un objet constant dans tout l'algorithme et ensuite de faire référence à cet objet par son nom plutôt que par sa valeur.

5.2.2 Les variables

Ce sont des « objets » dont la valeur peut changer au cours de l'exécution de l'algorithme.

Déclaration

Nom_Variable : type;

Exemple : x, y : nombres réels;

Déclarer une variable consiste à définir son type.

Exemple de types sous flowcode :

Bool	Booléen ou octet (nombre non signé à 8 bits)
Int	Entier, nombre signé à 16 bits entre -2^{15} et 2^{15} - 1, soit entre -32768 et 32767.
Float	Virgule flottante, nombre signé à 32 bits $\pm 2^{-149} \approx 1.4\times10^{-45} \rightarrow \pm 2^{128}-2^{104} \approx 3.4\times10^{38}$ soit 3.4 x 10^{-38} à 3.4 x 10^{+38})
Tableau	d'octets ou d'entiers à une dimension (nombre de cellules entre crochets)
Str	Chaine de caractère (Nombre de caractères entre crochets, 20 par défaut)

6. Système boucle ouverte –boucle fermé

Pilotage en boucle ouverte : La sortie dépend uniquement de la consigne d'entrée.

Problème : On ne tient pas compte des frottements et du couple résistant lié au coureur.

Pilotage en boucle fermée : La sortie dépend de la consigne de l'entrée mais aussi des éventuelles perturbations sur la sortie.